**ASYMMETRIC NUMERAL SYSTEMS: ADDING FRACTIONAL BITS TO HUFFMAN CODER**

### Data Compression

Data compression - used everywhere to store or transmit information

\[
\text{data} \xrightarrow{\text{statistical modelling}} \text{symbol sequence} (s_i, p_i) \xrightarrow{\text{entropy coder}} \text{bit sequence}
\]

**Huffman coding**

- *fast*, but operates on integer number of bits: approximates probabilities with powers of \( \frac{1}{2} \), getting inferior compression rate

**Arithmetic coding**

- *accurate*, but many times slower (computationally more demanding)

**Asymmetric Numeral Systems (ANS)**

- *accurate* and *faster* than Huffman
  
we construct low state automaton optimized for given probability distribution

---

Jarek Duda
Kraków, 28-04-2014
We need \( n \) bits of information to choose one of \( 2^n \) possibilities.

For length \( n \) 0/1 sequences with \( pn \) of “1”, how many bits we need to choose one?

\[
\binom{n}{pn} \approx 2^{nh(p)} \quad \text{for} \quad h(p) := -p \log_2(p) - (1-p) \log_2(1-p)
\]

**Entropy coder:** encodes sequence with \((p_i)_{i=1..m}\) probability distribution using asymptotically at least \( H = \sum_{i=1}^{m} p_i \log(1/p_i) \) bits/symbol \((H \leq \log(m))\)

Seen as weighted average: symbol of probability \( p \) contains \( \log(1/p) \) bits.

Encoding \((p_i)\) distribution with entropy coder optimal for \((q_i)\) distribution costs

\[
\Delta H = \sum_i p_i \log(1/q_i) - \sum_i p_i \log(1/p_i) = \sum_i p_i \log \left( \frac{p_i}{q_i} \right) \approx \frac{1}{\ln(4)} \sum_i \frac{(p_i - q_i)^2}{p_i}
\]

more bits/symbol - so called Kullback-Leibler “distance” (not symmetric).
Symbol frequencies from a version of the British National Corpus containing 67 million words (http://www.yorku.ca/mack/uist2011.html#f1):

**Uniform:** \( \lg(27) \approx 4.75 \text{ bits/symbol} \) \((x \to 27x + s)\)

**Huffman** uses: \( H' = \sum_i p_i r_i \approx 4.12 \text{ bits/symbol} \)

**Shannon:** \( H = \sum_i p_i \lg(1/p_i) \approx 4.08 \text{ bits/symbol} \)

We can theoretically improve by \( \approx 1\% \) here

\( \Delta H \approx 0.04 \text{ bits/symbol} \)

order 1 Markov: \( \sim 3.3 \text{ bits/symbol} \)

order 2: \( \sim 3.1 \text{ bits/symbol} \), word: \( \sim 2.1 \text{ bits/symbol} \)

Currently the best text compression: “durlica’kingsize” (http://mattmahoney.net/dc/text.html)

10^9 bytes of text from Wikipedia (enwik9) into 127784888 bytes: \( \approx 1 \text{ bit/symbol} \)

... (lossy) video compression: \( \sim 1000\times \) reduction
Huffman codes – encode symbols as bit sequences
Perfect if symbol probabilities are powers of 2
e.g. \( p(A) = 1/2, \ p(B) = 1/4, \ p(C) = 1/4 \)

We can reduce \( \Delta H \) by grouping \( m \) symbols together (alphabet size \( 2^m \) or \( 3^m \)):

\[
\Delta H = \frac{(1,1,1)/3}{m=2}
\]

\[\Delta H = \frac{2 \cdot 4 + 7 \cdot 3}{9} / 2 - \log(3) \approx 0.026 \text{ bits/symbol}\]

Generally, the maximal depth \( R = \max_i r_i \) grows proportionally to \( m \) here,
\( \Delta H \) drops approximately proportionally to \( 1/m \), \( \Delta H \propto 1/R \) \( (= 1/\log(L)) \)
for ANS: \( \Delta H \propto 4^{-R} \) \( (= 1/L^2 \text{ for } L = 2^R \text{ is the number of states}) \)
Fast Huffman decoding step for maximal depth $R$: let $X$ contain last $R$ bits (requires $decodingTable$ of size proportional to $L = 2^R$):

$t = decodingTable[X];$  \hfill // $X \in \{0, \ldots, 2^R - 1\}$ is current state
useSymbol($t$.symbol);  \hfill // use or store decoded symbol
$X = t.newX + readBits(t.nbBits);$  \hfill // state transition

where $t.newX$ for Huffman are unused bits of the state, shifted to oldest position:

$t.newX = (X \ll nbBits) \& (2^R - 1) \quad (\mod(a, 2^R) = a \& (2^R - 1))$

**tANS:** the same decoder, different $t.newX$: not only shifts the unused bits, but also modifies them accordingly to the remaining fractional bits ($\lg(1/p)$):

$x = X + L \in \{L, \ldots, 2L - 1\}$ buffer containing $\approx \lg(x) \in [R, R + 1)$ bits:

Encoding:

$s = a$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$Pr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.321</td>
</tr>
</tbody>
</table>

$s = b$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$Pr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.25</td>
</tr>
<tr>
<td>6</td>
<td>0.241</td>
</tr>
</tbody>
</table>

**Pr($a$) = 3/4**

Decoding:

$x \rightarrow s$, new $x$

<table>
<thead>
<tr>
<th>$x$</th>
<th>new $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$a$, 6+d$_1$</td>
</tr>
<tr>
<td>5</td>
<td>$b$, 4+2d$_2$+d$_1$</td>
</tr>
<tr>
<td>6</td>
<td>$a$, 4</td>
</tr>
<tr>
<td>7</td>
<td>$a$, 5</td>
</tr>
</tbody>
</table>

**Pr($b$) = 1/4**

$newX$, $nbBits$, $decodingTable$
Operating on fractional number of bits

We have information stored in a number $x$ and want to insert information of symbol $s=0,1$:

- **asymmetrize** ordinary/symmetric **binary system**: optimal for $\Pr(0)=\Pr(1)=1/2$

Most significant position: $x' = x + s \cdot 2^m$

Least significant position: $x' = 2x + s$

Restricting range $L$ to length $l$ subrange contains $\lg(L/l)$ bits

Adding symbol of probability $p$ - containing $\lg(1/p)$ bits

<table>
<thead>
<tr>
<th>$s=0$</th>
<th>$s=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s=0$</td>
<td>0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18...</td>
</tr>
<tr>
<td>$s=1$</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18...</td>
</tr>
</tbody>
</table>

**Range/Arithmetic coding:**

Rescale ranges

Restricting range $L$ to length $l$ subrange contains $\lg(L/l)$ bits

Number $x$ contains $\lg(x)$ bits

Adding symbol of probability $p$ - containing $\lg(1/p)$ bits

For $l' \approx p \cdot l$

For $x' \approx x/p$
Asymmetric numeral systems – redefine even/odd numbers:

symbol distribution:  \( \bar{s}: \mathbb{N} \to \mathcal{A} \)  
\( (\mathcal{A} – \text{alphabet, e.g. } \{0,1\}) \)
\( (\bar{s}(x) = \text{mod}(x, b) \text{ for base } b \text{ numeral system: } C(s, x) = bx + s) \)

Should be still uniformly distributed – but with density \( p_s \):

\[
\# \{ 0 \leq y < x: \bar{s}(y) = s \} \approx xp_s
\]

then \( x \) becomes \( x \)-th appearance of given symbol:

\[
D(x') = (\bar{s}(x'), \# \{0 \leq y < x': \bar{s}(y) = \bar{s}(x')\})
\]
\[
C(D(x')) = x' \quad D(C(s, x)) = (s, x) \quad x' \approx x/p_s
\]

example: range asymmetric binary systems (rABS)

\[
\text{Pr}(0) = 1/4 \quad \text{Pr}(1) = 3/4 \quad - \text{take base 4 system and merge 3 digits,}
\]
cyclic \( ([0123]) \) symbol distribution \( \bar{s} \) becomes cyclic \( ([0111]) \):

\[
\begin{array}{ccccccccccccccccccccccccccc}
x' & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 \\
(\bar{s},x) & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 & 4 & 4 \\
\end{array}
\]

\[
\bar{s}(x) = 0 \text{ if mod}(x, 4) = 0, \text{else } 1
\]
to decode or encode 1, localize quadruple \( ([x/4] \text{ or } [x/3]) \)

\[
\text{if } \bar{s}(x) = 0, D(x) = (0, [x/4]) \text{ else } D(x) = (1,3[x/4] + \text{mod}(x, 4) - 1)
\]
\[
C(0, x) = 4x \quad C(1, x) = 4|x/3| + \text{mod}(x, 3) + 1
\]
rANS - range variant for large alphabet \( \mathcal{A} = \{0, \ldots, m - 1\} \)

assume \( \Pr(s) = f_s/2^n \) \( c_s = f_0 + f_1 + \cdots + f_{s-1} \)

start with base \( 2^n \) numeral system and merge length \( f_s \) ranges

for \( x \in \{0,1, \ldots, 2^n - 1\} \), \( \bar{s}(x) = \max\{s: c_s \leq x\} \)

**encoding:** \( C(s, x) = \lfloor x/f_s \rfloor \ll n + \text{mod}(x, f_s) + c_s \)

**decoding:** \( s = \bar{s}(x \& (2^n - 1)) \) (e.g. tabled, alias method)

\( D(x) = \left( s, f_s \cdot (x \gg n) + (x \& (2^n - 1)) - c_s \right) \)

Similar to Range Coding, but decoding has 1 multiplication (instead of 2), and state is 1 number (instead of 2), making it convenient for SIMD vectorization. (https://github.com/ryg_rans).

Additionally, we can use alias method to store \( \bar{s} \) for very precise probabilities. It is also convenient for dynamical updating.

‘Alias’ method: rearrange probability distribution into \( m \) buckets: containing the primary symbol and eventually a single ‘alias’ symbol
uABS - uniform binary variant \((\mathcal{A} = \{0,1\})\) - extremely accurate

Assume binary alphabet, \(p := \text{Pr}(1)\), denote \(x_s = \{y < x: \bar{s}(y) = s\} \approx xp_s\)

For uniform symbol distribution we can choose:

\[
x_1 = \lfloor xp \rfloor \quad x_0 = x - x_1 = x - \lfloor xp \rfloor
\]

\(\bar{s}(x) = 1\) if there is jump on next position:

\[
s = \bar{s}(x) = \lfloor (x + 1)p \rfloor - \lfloor xp \rfloor
\]

developing function: \(D(x) = (s, x_s)\)

its inverse – coding function:

\[
C(0, x) = \left\lfloor \frac{x+1}{1-p} \right\rfloor - 1
\]

\[
C(1, x) = \left\lfloor \frac{x}{p} \right\rfloor
\]

For \(p = \text{Pr}(1) = 1 - \text{Pr}(0) = 0.3:\)

\[
C(s,x) \approx \frac{x}{\text{Pr}(s)}
\]

| \(C(s,x)\) | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  |
|------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| \(x\)     | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 | s=0 |
|            | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  |
| \(px\)    | 0.0 | 0.9 | 1.8 | 3.0 | 3.9 | 4.8 | 6.0 | 6.9 | 7.2 | 8.1 | 9.0 | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  |
Stream version – renormalization

Currently: we encode using succeeding $C$ functions into huge number $x$, then decode (in reverse direction!) using succeeding $D$.

Like in arithmetic coding, we need renormalization to limit working precision - enforce $x \in I = \{L, \ldots, bL - 1\}$ by transferring base $b$ youngest digits:

<table>
<thead>
<tr>
<th>ANS <strong>decoding</strong> step from state $x$</th>
<th><strong>encoding</strong> step for symbol $s$ from state $x$</th>
</tr>
</thead>
</table>
| $(s, x) = D(x);$ useSymbol($s$); while $x < L$, $x = bx + \text{readDigit}();$ | while $x > \text{maxX}[s] \quad // = bL_s - 1$
{\text{writeDigit(mod($x, b$)); $x = \lfloor x/b \rfloor$};
$x = C(s, x);$ |

For unique decoding, we need to ensure that there is a single way to perform above loops:

$I = \{L, \ldots, bL - 1\}$,  $I_s = \{L_s, \ldots, bL_s - 1\}$ where $I_s = \{x: C(s, x) \in I\}$

Fulfilled e.g. for
- rABS/rANS when $p_s = f_s/2^n$ has $1/L$ accuracy: $2^n$ divides $L$,
- uABS when $p$ has $1/L$ accuracy: $b[Lp] = [bLp]$,
- in tabled variants (tABS/tANS) it will be fulfilled by construction.
Single step of stream version:

to get $x \in I$ to $I_s = \{L_s, \ldots, bL_s - 1\}$, we need to transfer $k$ digits:

$$x \rightarrow \left( C(s, \lfloor x/b^k \rfloor), \mod(x, b^k) \right)$$

where $k = \lfloor \log_b(x/L_s) \rfloor$

for given $s$, observe that $k$ can obtain one of two values:

$k = k_s$ or $k = k_s - 1$ for $k_s = -\lfloor \log_b(p_s) \rfloor = -\lfloor \log_b(L_s/L) \rfloor$

e.g.: $b = 2, k_s = 3, L_s = 13, L = 66, b^{k_s}x + 3 = 115, x = 14, p_s = 13/66$: 

\[ L \rightarrow \text{transfer of } k_s-1 \text{ digits} \]

\[ C(s, L_s) \rightarrow C(s, x) \approx \frac{L}{p_s b^{k_s-1}} \]

\[ X_s = L_s b^{k_s} \approx L p_s b^{k_s} \rightarrow \text{transfer of } k_s \text{ digits} \]

\[ b^{k_s} x + 3 \]

\[ \approx \frac{1}{p_s} \]

\[ C(s, L_s b-1) \rightarrow \text{stream decoding} \]
**General picture:**
encoder prepares before consuming succeeding symbol
decoder produces symbol, then consumes succeeding digits

- **Decoding is in reversed direction:** we have stack of symbols (LIFO)
  - the final state has to be stored, but we can write information in initial state
  - encoding should be made in backward direction, but use context from perspective of future forward decoding.
In single step \((I = \{L, \ldots, bL - 1\})\): \(\lg(x) \rightarrow \approx \lg(x) + \lg(1/p) \mod \lg(b)\)

Three sources of **unpredictability/chaosity**:

1) **Asymmetry**: behavior strongly dependent on chosen symbol – small difference changes decoded symbol and so the entire behavior.

2) **Ergodicity**: usually \(\log_b(1/p)\) is irrational – succeeding iterations cover entire range.

3) **Diffusivity**: \(C(s, x)\) is close but not exactly \(x/p_s\) – there is additional ‘diffusion’ around expected value

So \(\lg(x) \in [\lg(L), \lg(L) + \lg(b))\) has nearly uniform distribution – \(x\) has approximately:

\[
\Pr(x) \propto 1/x
\]

probability distribution – contains \(\lg(1/\Pr(x)) \approx \lg(x) + \text{const}\) bits of information.
Redundancy: instead of $p_s$ we use $q_s = x / C(s, x)$ probability:

$$
\Delta H \approx \frac{1}{\ln(4)} \sum_s \frac{(p_s - q_s)^2}{p_s}
$$

... $$
\Delta H \approx \frac{1}{\ln(4)} \sum_{s,x} \Pr(x) \frac{(\epsilon_s(x))^2}{p_s}
$$

where inaccuracy: $\epsilon_s(x) = p_s - x / C(s, x)$ drops like $1/L$:

$$
\Delta H \leq \frac{1}{\ln(4) \cdot L^2} \left( \frac{1}{p} + \frac{1}{1-p} \right) \text{ for uABS, } \Delta H \leq \frac{m}{\ln(4) \cdot L^2} \sum_s \frac{1}{p_s} \text{ for rANS}
$$

Denoting $L = 2^R$, $\Delta H \approx 4^{-R}$ and grows proportionally to $(\text{alphabet size})^2$.

for uABS:
Tabled variants: tANS, tABS  (choose $L = 2^R$)

$I = \{L, \ldots, 2L - 1\}$, $I_s = \{L_s, \ldots, 2L_s - 1\}$  where $I_s = \{x: C(s, x) \in I\}$

we will choose $\overline{s}$ symbol distribution (symbol spread):\#

$\{x \in I: \overline{s}(x) = s\} = L_s$  approximating probabilities: $p_s \approx L_s/L$

Fast pseudorandom spread (https://github.com/Cyan4973/FiniteStateEntropy):

\begin{enumerate}
    \item $step = 5/8 \cdot L + 3$;  // step chosen to cover the whole range\item $X = 0$;  // $X = x - L \in \{0, \ldots, L - 1\}$  for table handling\end{enumerate}

for $s = 0$ to $m - 1$ do
\begin{enumerate}
    \item $\{symbol[X] = s; X = \text{mod}(X + step, L);\}$  // $symbol[X] = \overline{s}(X + L)$\end{enumerate}

Then finding table for\begin{enumerate}
    \item $t = \text{decodingTable}(X)$; $\text{useSymbol}(t.\text{symbol})$;\item $X = t.\text{newX} + \text{readBits}(t.\text{nbBits})$\end{enumerate}

for $s = 0$ to $m - 1$ do $next[s] = L_s$;  // symbol appearance number
for $X = 0$ to $L - 1$ do  // fill all positions
\begin{enumerate}
    \item $t.\text{symbol} = symbol[X]$;  // from symbol spread\item $x = next[t.\text{symbol}] + +$;  // use symbol and shift appearance\item $t.\text{nbBits} = R - \lfloor \lg(x) \rfloor$;  // $L = 2^R$\item $t.\text{newX} = (x \ll t.\text{nbBits}) - L$; $\text{decodingTable}[X] = t;$\end{enumerate}
Precise initialization (heuresis)\n
\[ N_s = \left\{ \frac{0.5+i}{p_s} : i = 0, \ldots, L_s - 1 \right\} \]

are uniform – we need to shift them to natural numbers.

(priority queue with put, getminv)

\[
\begin{align*}
&\text{for } s = 0 \text{ to } n - 1 \text{ do} \\
&\quad \text{put}((0.5/p_s, s)); \\
&\text{for } X = 0 \text{ to } L - 1 \text{ do} \\
&\quad ((v, s) = \text{getminv}; \\
&\quad \text{put}((v + 1/p_s, s)); \\
&\quad \text{symbol}[X] = s; \\
&\end{align*}
\]

\[ \Delta H \text{ drops like } 1/L^2 \]

\[ L \propto \text{alphabet size} \]
tABS and tuning

test all possible symbol distributions for binary alphabet

store tables for quantized probabilities ($p$)

e.g. $16 \cdot 16 = 256$ bytes

← for 16 state, $\Delta H \approx 0.001$

---

**H.264 “M decoder” (arith. coding)**

```c
// interval subdivision
1:   \( R_{\text{lps}} = \text{RTAB}[m] [(R >> 6) \& 3] \)
2:   \( R_{\text{mps}} = R - R_{\text{lps}} \)
3:   if (\( V < R_{\text{mps}} \))
4:       \( R = R_{\text{mps}}, \text{ value } = \text{valMPS} \)
5:   else
6:       \( V = V - R_{\text{mps}}, \text{ value } = !\text{valMPS} \)
7:   \( R = R_{\text{lps}} \)

// renormalization
8:   while (\( R < 2^x \))
9:       \( R = R << 1 \)
10:  \( V = V << 1 \)
11:  \( V = V | \text{read_one_bit()} \)
```

---

tABS

\[
\begin{align*}
t & = \text{decodingTable}[p][X]; \\
X & = t.\text{newX} + \text{readBits}(t.\text{nbBits}); \\
\text{useSymbol}(t.\text{symbol});
\end{align*}
\]

no branches,

no bit-by-bit renormalization

state is single number (e.g. SIMD)
**Additional tANS advantage – simultaneous encryption**

we can use huge freedom while **initialization**: choosing symbol distribution – slightly disturb $\bar{s}(x)$ using PRNG initialized with cryptographic key

**ADVANTAGES comparing to standard (symmetric) cryptography:**

<table>
<thead>
<tr>
<th></th>
<th>standard, e.g. DES, AES</th>
<th>ANS based cryptography (<strong>initialized</strong>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>based on</td>
<td>XOR, permutations</td>
<td>highly <strong>nonlinear</strong> operations</td>
</tr>
<tr>
<td>bit blocks</td>
<td>fixed length</td>
<td>pseudorandomly <strong>varying lengths</strong></td>
</tr>
<tr>
<td>“brute force” or QC attacks</td>
<td>just start decoding to test cryptokey</td>
<td><strong>perform initialization first</strong> for new cryptokey, fixed to need e.g. 0.1s</td>
</tr>
<tr>
<td>speed</td>
<td>online calculation</td>
<td>most calculations while initialization</td>
</tr>
<tr>
<td>entropy</td>
<td>operates on bits</td>
<td>operates on <strong>any input distribution</strong></td>
</tr>
</tbody>
</table>

**initialized** - most of calculations made while **unavoidable** initialization

(if decoded message makes sense)

some limit to test cryptokey in **brute force attack**

time to reach this limit while testing succeeding cryptokeys, or calculations to maintain quantum entanglement of cryptokeys in hypothetical QC attack
Summary – we can construct accurate and extremely fast entropy coders:
- accurate (and faster) replacement for Huffman coding,
  - many times faster replacement for Range coding,
- faster decoding in adaptive binary case (but more complex encoding),
  - can simultaneously encrypt the message.

Perfect for: DCT/wavelet coefficients, lossless image compression
Perfect with: Lempel-Ziv, Burrows-Wheeler Transform
... and many others ...

Further research:
- finding symbol distribution $\bar{s}(x)$: with minimal $\Delta H$ (and quickly),
- tune accordingly to $p_s \approx L_s/L$ approximation (e.g. very low probable symbols should have single appearance at the end of table),
- optimal compression of used probability distribution to reduce headers,
- maybe finding other low state entropy coding family, like forward decoded,
  - applying cryptographic capabilities (also without entropy coding),
  - ...?